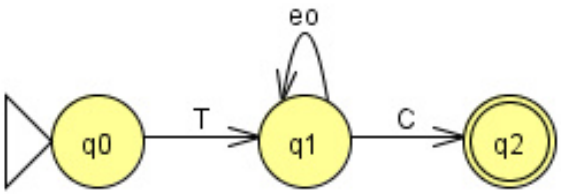
UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
Departamento de Engenharia Informática

Teoria da Computação

2006/2007

Trabalho prático nº 1



Trabalho realizado por:

Pedro Oliveira (501062444)
Rui Costa (501062452)

Turma: TP1

1 - Introdução

O objectivo deste trabalho era implementar um simulador de Autómatos Finitos Determinísticos. Dentro deste objectivo havia várias etapas que eram pedidas e foram implementadas, sendo elas:

- Ser possível ao utilizador introduzir um autómato;
- Ler e gravar um autómato de e para um ficheiro;
- Rejeitar autómatos inválidos (p.e. autómato que não sejam deterministas);
- Determinar se um autómato aceita uma cadeia de caracteres;
- Simular passo a passo o autómato para diversas cadeias de caracteres;

O trabalho foi começado por ser implementado em duas linguagens de programação distintas: *Java* e *Python*. Tendo sido escolhida a implementação em *Python* por se ter revelado mais aliciante para nós que desconhecíamos a linguagem e porque se veio a revelar uma linguagem extremamente eficaz para este trabalho em específico.

2 – Manual do utilizador

Neste manual de utilizador será explicada a forma como o utilizador deve proceder de forma a interagir correctamente com o programa. O menu contém diversas opções que são explicadas de seguida.

1 - Introduzir Autómato

Opção que permite introduzir um autómato, as opções Simular Autómato e Guardar Autómato em ficheiro têm como pré-requisito a correcta execução desta opção (ou a opção "Ler Autómato de Ficheiro").

O que é	Input	notas
Estados(Q)	q0, q1	Os estados devem estar separados por caracteres que não de (A-Z) e (0-9).
Alfabeto(∑)	{a,b}	Não é relevante a forma como é inserido desde que não estejam separados por caracteres de (A-Z) e (0-9).
Transições(δ)	q0:a->q1; q1:a->q1;	Devem obedecer à representação descrita no exemplo (uma transição por linha). A inserção de transições termina quando introduzida uma linha em branco. E é após esta que é avaliado o determinismo do autómato.
Estado inicial(q_0)	q0	É constituído por apenas um estado.
Estados finais(F)	q1, q0	Os estados devem estar separados por caracteres que não de (A-Z) e (0-9).

2 – Ler Autómato de ficheiro

Opção que permite carregar um autómato previamente guardado num ficheiro.

O que é	Input	notas
Nome do ficheiro	t1.txt	O conteúdo do ficheiro deve ser igual ao que é mostrado no tópico introduzir autómato e o ficheiro deve terminar com um \n.

3 – Guardar Autómato em ficheiro

Permite guardar um autómato finito determinístico gerado em um ficheiro.

O que é	Input	notas
Nome do ficheiro	t1.txt	-

4 – Simular Autómato

Permite simular uma cadeia de símbolos de forma a verificar se a mesma é ou não aceite pelo autómato inserido. Esta simulação pode ser feita passo a passo (i.e. um caracter de cada vez) sendo apenas necessário pressionar o enter para que se avance para o próximo passo. A outra forma de proceder à avaliação transita automaticamente para o final, sabendo o utilizador se a cadeia é ou não aceite.

O que é	Input	notas
Cadeia	aa	-
Tipo de simulação	1 ou 2	Automática ou passo a passo.

3 – Manual do programador

Na elaboração deste projecto, escolhemos usar a linguagem *Python*. Foi um desafio em duas vertentes: o desenvolvimento do projecto em si e a aprendizagem de uma nova linguagem.

No final do desenvolvimento da aplicação, chegámos à conclusão que a escolha do *Python* mostrou-se muito lucrativa, visto que esta linguagem possui características que mostraram ser muito vantajosas para este tipo de projectos. A potência das suas listas e dicionários, aliados á possibilidade de criação de tuplos de variáveis permitiram-nos reduzir em muito a complexidade da aplicação em comparação com outras linguagens como C e Java.

Como a interface não era valorizada, a interacção com o utilizador é efectuada num sistema simples de consola.

3.1 Gramática Utilizada

A gramática utilizada foi baseada na referida no enunciado com pequenas modificações que, para nós, melhoram a interacção com o utilizador.

Exemplo de Automato:

```
q0, q1
{a,b,c}
q0:a->q1, b->q1;
q1:a->q0, b->q0;

q0
q0, q1
```

A inserção de estados, alfabeto, estado inicial e estados finais é igual à referida no enunciado. Apenas as transições são diferentes. Decidimos que era mais fácil para o utilizador inserir as transições de cada estado numa linha só, para uma melhor ordenação e distribuição do input. O input de transições acaba com uma linha em branco.

Todos os caracteres “especiais” que estejam fora do nosso abecedário (a-Z) ou da nossa numeração (0-9) são ignorados do input, logo não é, por exemplo, obrigatório separar a inserção de estados por uma vírgula. Pode-se pura e simplesmente inserir um espaço ou outro carácter qualquer que esteja fora dos alfabetos referidos.

3.2 Estruturas de dados

Para representar o autómato recorremos a estruturas de dados características do *Python*. O autómato em si é um objecto (AFD) que contem essas estruturas de dados.

Classe AFD

```
estados = []
alfabeto = []
transicoes = {}
estado_inicial = ""
estados_finais = []
loaded = 0
```

Traduzindo para a definição de autómato finito (Q, Σ, δ, S, F):

Q = estados[]
Σ = alfabeto[]
δ = transicoes{}
S = estado_inicial
F = estados_finais[]

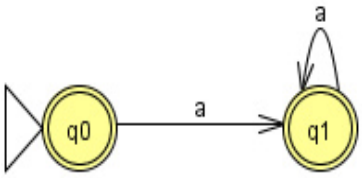
Os estados, alfabeto e estados finais são representados numa lista. O estado inicial é uma simples *string*. As transições são representadas por um dicionário, cuja chave é um tuplo (estado,letra) e conteúdo é o estado de destino. A variável *loaded* serve apenas para confirmar se o autómato é correcto e/ou determinístico.

3.3 Testes

Foram elaborados vários testes para avaliar a eficácia e qualidade da aplicação. Depois de efectuados esses testes não detecta-mos nenhuma anomalia, por isso a avaliação feita por nós à aplicação foi muito satisfatória.

Como exemplo, seguem alguns dos testes efectuados.

Teste1

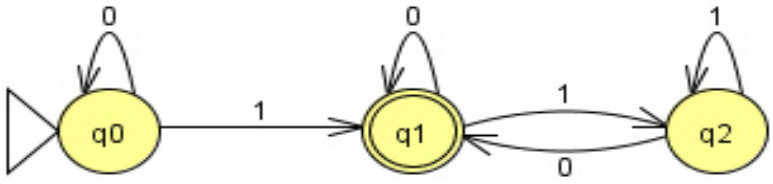



```
q0, q1
{a}
q0:a->q1;
q1:a->q1;

q0
q0, q1
```

Input	Resultado Esperado	Resultado Obtido
‘’	q0 -> Aceite	q0 -> Aceite
‘aaaaaa’	q1 -> Aceite	q1 -> Aceite

Teste2

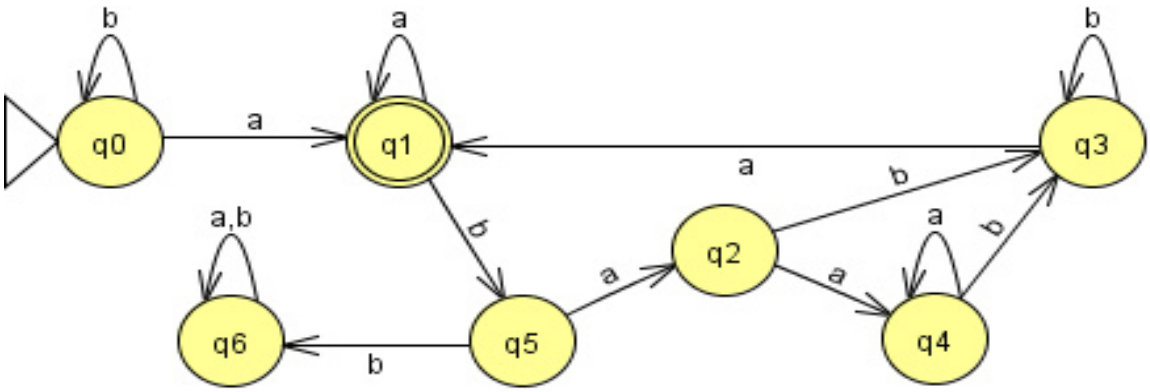



```
q0, q1, q2
{0,1}
q0:0->q0, 1->q1;
q1:0->q1, 1->q2;
q2:0->q1, 1->q1;

q0
q1
```

Input	Resultado Esperado	Resultado Obtido
‘’	q0 -> Rejeitada	q0 -> Rejeitada
‘0’	q0 -> Rejeitada	q0 -> Rejeitada
‘00000110’	q1 -> Aceite	q1 -> Aceite
‘1111111100001010’	q1 -> Aceite	q1 -> Aceite
‘1111111111111111’	q2 -> Rejeitada	q2 -> Rejeitada

Teste3

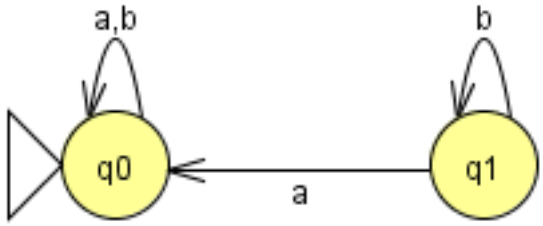



```
q0, q1, q2, q3, q4, q5, q6
{a,b}
q0:a->q1, b->q0;
q1:a->q1, b->q5;
q2:a->q4, b->q3;
q3:a->q1, b->q3;
q4:a->q4, b->q3;
q5:a->q2, b->q6;
q6:a->q6, b->q6;

q0
q1
```

Input	Resultado Esperado	Resultado Obtido
‘’	q0 -> Rejeitada	q0 -> Rejeitada
‘ababa’	q1 -> Aceite	q1 -> Aceite
‘bbaaabababbbbbaaaa’	q6 -> Rejeitada	q6 -> Rejeitada
‘ababaaaabaaaaabbbbaaaaa’	q1 -> Aceite	q1 -> Aceite
‘baabbaababababaaababababababababaaaaab’	q6 -> Rejeitada	q6 -> Rejeitada

Teste 4

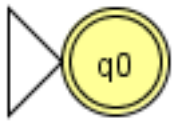



```
q0, q1
{a,b}
q0:a->q0, b->q0;
q1:a->q0, b->q1;

q0
```

Input	Resultado Esperado	Resultado Obtido
‘’	q0 -> Rejeitada	q0 -> Rejeitada
‘ababa’	q0 -> Rejeitada	q0 -> Rejeitada

Teste 5




```
q0


q0
q0
```

Input	Resultado Esperado	Resultado Obtido
‘’	q0 -> Aceite	q0 -> Aceite